Beschreibung

Verknüpfung und Darstellung von Signalen einer Vorrichtung zur Hardware-Simulation und Elementen eines Listings eines Programms

Die Erfindung betrifft ein System sowie ein Verfahren zur Darstellung von Signalen einer Vorrichtung zur Hardware-Simulation sowie ein Fehlersuchwerkzeug.

Aus US 5 768 567 ist ein Hardware-Software-Cosimulator zur Simulation eines Hardware-Software-Systems bekannt, welcher einen logischen Simulator, Busschnittstellenmodelle, Speichermodelle, Befehlssatzsimulatoren und einen sogenannten Cosimulations-Optimierungs-Manager aufweist. Die gleichzeitige Simulation von Hardware und Software wird auch Cosimulation genannt. Dabei wird die Cosimulation mit einem einzigen kohärenten Blick auf den Speicher des Hardware-Software-Systems durchgeführt, welcher durch den Cosimulations-Optimierungs-Manager sowohl für Hardware- als auch für Software-Simulationen transparent aufrechterhalten wird.

Der Erfindung liegt die Aufgabe zugrunde, die gemeinsame Darstellung von Signalen einer Vorrichtung zur Hardware-Simulation und Elementen eines Listings eines Programms zu ermöglichen.

Diese Aufgabe wird durch ein System zur Verknüpfung und Darstellung von Signalen einer Vorrichtung zur Hardware-Simulation und Elementen eines Listings eines Programms gelöst,

- wobei die Vorrichtung zur Hardware-Simulation das Verhalten einer Schaltung mit einem Prozessor, einem Programmspeicher, welcher den Programmcode des Programms enthält, und applikationsspezifischen Hardwarekomponenten simuliert und Signale als Ergebnis der Simulation erzeugt,

- wobei die Elemente des Listings des Programms mit den bei der simulierten Ausführung des im Programmspeicher enthaltenen, mit diesen Elementen korrespondierenden Programmcodes erzeugten Signalen verknüpft werden,
- wobei die Elemente des Listings des Programms in einem ersten Teilbereich eines grafischen Anzeigemittels und die Signale in einem zweiten Teilbereich des Anzeigemittels darstellbar sind.

Diese Aufgabe wird durch ein Verfahren zur Verknüpfung und Darstellung von Signalen einer Vorrichtung zur Hardware-Simulation und Elementen eines Listings eines Programms gelöst,

- wobei die Vorrichtung zur Hardware-Simulation das Verhalten einer Schaltung mit einem Prozessor, einem Programmspeicher, welcher den Programmcode des Programms enthält, und applikationsspezifischen Hardwarekomponenten simuliert und Signale als Ergebnis der Simulation erzeugt,
- wobei die Elemente des Listings des Programms mit den bei der simulierten Ausführung des im Programmspeicher enthaltenen, mit diesen Elementen korrespondierenden Programmcodes erzeugten Signalen verknüpft werden,
- wobei die Elemente des Listings des Programms in einem ersten Teilbereich eines grafischen Anzeigemittels und die Signale in einem zweiten Teilbereich des Anzeigemittels dargestellt werden.

Diese Aufgabe wird durch ein Fehlersuchwerkzeug zur Verknüpfung und Darstellung von Signalen einer Vorrichtung zur Hardware-Simulation und Elementen eines Listings eines Programms gelöst,

- wobei die Vorrichtung zur Hardware-Simulation das Verhalten einer Schaltung mit einem Prozessor, einem Programmspeicher, welcher den Programmcode des Programms enthält, und applikationsspezifischen Hardwarekomponenten simuliert und Signale als Ergebnis der Simulation erzeugt,

- wobei das Fehlersuchwerkzeug Mittel zur Verknüpfung der Elemente des Listings des Programms mit den bei der simulierten Ausführung des im Programmspeicher enthaltenen, mit diesen Elementen korrespondierenden Programmcodes erzeugten Signalen aufweist,

wobei die Elemente des Listings des Programms in einem ersten Teilbereich eines grafischen Anzeigemittels und die Signale in einem zweiten Teilbereich des Anzeigemittels darstellbar sind.

Der Erfindung liegt die Erkenntnis zugrunde, dass das Design von Hardware-Software-Systemen durch eine verknüpfte Darstellung der Signale eines Hardware-Simulators und der Elemente des Listings eines Programms wesentlich vereinfacht wird. Das erfindungsgemäße System und Verfahren ermöglicht eine einfache und fehlersichere Verfolgung der Ausführung eines Programms. Die Darstellung des Listings des Programms stellt einen direkten Bezug zum eigentlichen Programm her, insbesondere können auch aufgrund der Verwendung des Listfiles die im Original-Quelltext eingefügten Kommentare angezeigt werden. Zur Verknüpfung und Darstellung der Signale der Vorrichtung zur Hardware-Simulation und der Elemente des Listings des Programms ist das Vorliegen eines abstrahierten Software-Modells des Prozessors nicht erforderlich. Es ist somit gewährleistet, dass tatsächlich der in der Schaltung verwendete Prozessor simuliert wird. Somit werden durch eine Beschreibung des Prozessors mittels eines Modells verursachte Fehler bei der Simulation ausgeschlossen. Die Visualisierung des Programmablaufs erfolgt in ähnlicher Form wie bei üblichen Entwicklungswerkzeugen zum Debuggen von reiner Software. Die Einarbeitungszeit für Anwender des Systems bzw. des Verfahrens, welche Erfahrung bei der Entwicklung von Software oder Firmware haben, ist daher relativ gering.

Gemäß einer vorteilhaften Ausgestaltung der Erfindung ist eine Markierung eines Elements des Listings des Programms im ersten Teilbereich des grafischen Anzeigemittels und eine

Markierung der mit diesem Element verknüpften Signale im zweiten Teilbereich des Anzeigemittels vorgesehen. Die grafische Darstellung orientiert sich somit an üblichen Software-Debug-Werkzeugen. Der Programmverlauf kann durch eine Markierung eines Elements des Listings des Programms verfolgt werden. Im zweiten Teilbereich des Anzeigemittels bewegt sich eine Markierung synchronisiert zur Markierung im ersten Teilbereich des Anzeigemittels, so dass auch Querbeziehungen zur übrigen Hardware erfasst werden können.

Vorteilhafterweise ist ein dritter Teilbereich des grafischen Anzeigemittels zur Darstellung mindestens eines Teils der Signale, insbesondere weiterer Werte, vorgesehen. Weitere Werte können beispielsweise Registerwerte sein.

Die Verwendung üblicher Vorrichtungen zur Hardware-Simulation wird erleichtert, wenn gemäß einer vorteilhaften Ausgestaltung der Erfindung die Schaltung mit dem Prozessor, dem Programmspeicher und den applikationsspezifischen Hardwarekomponenten in einer Hardware-Beschreibungssprache beschrieben sind.

Das System lässt sich mit geringem Aufwand an verschiedene Prozessoren anpassen, wenn gemäß einer vorteilhaften Ausgestaltung der Erfindung Mittel zum Anpassen des Systems an verschiedene Prozessortypen vorgesehen sind.

Nachfolgend wird die Erfindung anhand der in den Figuren dargestellten Ausführungsbeispiele näher beschrieben und erläutert.

Es zeigen:

FIG 1 eine schematische Darstellung eines Systems zur Verknüpfung und Darstellung von Signalen einer Vorrichtung zur Hardware-Simulation und Elementen eines Listings eines Programms,

FIG 2 einen Ausschnitt einer Darstellung von Signalen einer Vorrichtung zur Hardware-Simulation und

FIG 3 eine Darstellung von Signalen einer Vorrichtung zur Hardware-Simulation und Elementen eines Listings eines Programms.

FIG 1 zeigt ein System zur Verknüpfung und Darstellung von Signalen 4, 5 einer Vorrichtung zur Hardware-Simulation 1 und Elementen 15 eines Listings 2 eines Programms 3 in schematischer Darstellung. Die Vorrichtung zur Hardware-Simulation 1 simuliert das Verhalten einer Schaltung 6 mit einem Prozessor 7, einem Programmspeicher 8 und applikationsspezifischen Hardwarekomponenten 10. Der Prozessor 7 kann z. B. ein Mikroprozessor oder Mikrocontroller sein. Der Programmspeicher 8 enthält den Programmcode 9 des Programms 3. Die Schaltung 6 wird mit Hilfe einer Hardware-Beschreibungssprache, abgekürzt HDL (HDL = Hardware Description Language), im Ausführungsbeispiel mittels VHDL (VHDL = Very High Speed Integrated Circuit Hardware Description Language) beschrieben. Mit dem Bezugszeichen 17 wird der Quellcode in VHDL bezeichnet. Die Vorrichtung zur Hardware-Simulation 1 wird entsprechend auch als HDL-Simulator bezeichnet. Ein Beispiel für einen HDL-Simulator ist das Produkt "ModelSim" der Firma Model Technology, Portland, Oregon, USA. Die Schaltung 6 ist z. B. ein sogenannter ASIC (ASIC = Application Specific Integrated Circuit = Anwendungsspezifische integrierte Schaltung). Der HDL-Quellcode 17 wird mit einem geeigneten Compiler in ein HDL-Simulator-Format 18 umgewandelt, welches von der Vorrichtung zur Hardware-Simulation 1 verarbeitet werden kann. Die Vorrichtung zur Hardware-Simulation 1 erzeugt Signale 4, 5 als Ergebnis der Simulation. Das Programm 3, bzw. der Programmcode 9 des Programms 3 wird mittels eines Compilers in ein Datenfile 16 (üblicherweise mit Daten in hexadezimaler Form) und ein Listing 2, auch Listfile genannt, umgewandelt. Das Listing 2 enthält sowohl die Programmbefehle als auch die zu-

gehörigen Kommentare. Das Listing 2 ist zeilenweise aufgebaut, wobei jede Zeile jeweils einen Programmbefehl bzw. eine Anweisung enthält, gegebenenfalls mit dem zugehörigen Kommentar. Eine Zeile, ein Programmbefehl, eine Anweisung bzw. ein Kommentar sind Elemente 15 des Listings 2. Ein Debugger 19 verknüpft die Elemente 15 des Listings 2 des Programms 3 mit den bei der simulierten Ausführung des im Programmspeicher 8 enthaltenen, mit diesen Elementen 15 korrespondierenden Programmcodes 9 erzeugten Signalen 4, 5. Die Elemente 15 des Listings 2 des Programms 3 werden in einem ersten Teilbereich 11 des grafischen Anzeigemittels 14 und die Signale 4, 5 in einem zweiten Teilbereich 12 bzw. in einem dritten Teilbereich 13 des Anzeigemittels 14 dargestellt.

FIG 2 zeigt einen Ausschnitt 30 einer Darstellung von Signalen 4, 5 einer Vorrichtung zur Hardware-Simulation 1. Die Signale 4, 5 werden als sogenannte Waveforms 35, 36 und 37 dargestellt.

FIG 3 zeigt eine Darstellung 23 von Signalen 4, 5 einer Vorrichtung zur Hardware-Simulation 1 und Elementen 15 eines Listings 2 eines Programms 3. Die Elemente 15 des Listings 2 werden in einem ersten Teilbereich 11, die Signale 4, 5 in einem zweiten 12 bzw. einem dritten 13 Teilbereich eines Anzeigemittels 14 dargestellt. Die Elemente 15 können mit einer Markierung 20, z. B. einer farblichen Kennzeichnung, versehen werden. Die Signale 4 können mit einer Markierung 21, z. B. einem Cursor, gekennzeichnet werden. Ein Anzeigemittel 14 kann eine Bildschirmoberfläche sein, die Teilbereiche 11 bis 13 können als Anzeigefenster realisiert sein. Der Ablauf des Programms 3 im Prozessor 7 (siehe FIG 1) wird mit Hilfe des grafischen Frontends, des Debuggers 19, welcher auf die Signale 4, 5 der Vorrichtung zur Hardware-Simulation 1 aufsetzt, visualisiert. Die Visualisierung des Programmablaufes erfolgt durch Zugriff auf Signale des Prozessors 7, deren Werte durch die Vorrichtung zur Hardware-Simulation 1 berechnet wurden.

Gemäß dem Ausführungsbeispiel ist der Debugger 19 in der Skriptsprache TCL/TK realisiert (TCL/TK = Tool Command Language/Toolkit) und benutzt die TCL/TK-Schnittstelle zu HDL-Objekten, die ein HDL-Simulator zur Verfügung stellt. Solche HDL-Objekte können z. B. als Waveforms dargestellt werden. Um den Debugger 19 möglichst flexibel zu gestalten, d. h. in diesem Zusammenhang, dass er relativ einfach an verschiedene Prozessoren angepasst werden kann, wird der Debugger 19 in zwei Abschnitte aufgeteilt. Ein erster allgemeiner Teil stellt Prozeduren zur Visualisierung bereit. Ein zweiter prozessorspezifischer Teil ist dem jeweiligen Prozessor anpassbar und setzt sich z. B. aus folgenden Prozeduren zusammen:

- Prozeduren für Single-Step rechts/links
- Prozeduren zum Bestimmen der Registerwerte
- Prozeduren für die Kopplung von Signalen und Elementen des Listings
- Prozeduren zur Konfiguration

In den Prozeduren für Single-Step rechts/links (Bezugszeichen 22 in FIG 3) wird der Cursor in einem Waveform-Fenster auf den nächsten gültigen Befehl gesetzt. Für das Beispiel eines KRISC8-Prozessors (KRISC8 = Kommunikations-RISC 8 Bit, Spezial-Core für die Kommunikation in Feldbussystemen der Firma Siemens AG, München, Deutschland) ist dies in FIG 2 veranschaulicht.

Der im Folgenden wiedergegebene Ausschnitt aus einem Listing zeigt eine Prozedur für Single-Step-Right.

```
# Prozedur für Single-Step-Right
# ACHTUNG: pc muss im Wave-Fenster selektiert sein (Das right-Kommando wirkt auf das
#          selektierte Signal)!!!!!
proc right_krisc8.Proc {} {
   global minideb
   global DEBUG
   set time_now [lindex [getactivecursortime] 0]
   set address [examine -time $time_now $minideb(TIME_UNIT) -hex $minideb(pc-path)]
   set address_tmp [examine -time [expr $time_now + $minideb(CLK_PERIOD)] $minideb(TIME_UNIT) -hex
$minideb(pc-path)]
   set i 1
   if {$DEBUG == "on"} {echo "anfang von right.Proc"}
   if {$address_tmp == "No_Data"} {
      .mainFrame.label_message configure -bg DimGray -fg red -text "End of Simulation reached."
      set minideb(auto_step_right) 0
      .mainFrame.frame_toolbar.autostep_right configure -background grey -activebackground khaki2
      .mainFrame.frame_toolbar.label_info configure -text ""
      return
   }
   # Anfang vom naechsten Befehl suchen
   while {$address == $address_tmp} {
      if {$DEBUG == "on"} {echo "right.Proc: while-Schleife 1"}
      incr i
      set address_tmp [examine -time [expr $time_now + $i * $minideb(CLK_PERIOD)] $minideb(TIME_UNIT)
-hex $minideb(pc-path)]
   }
   set time_now [expr $time_now + $i * $minideb(CLK_PERIOD)]
   if {$DEBUG == "on"} {echo "right.Proc:time_now --> $time_now"}
   set schleife 1
   # Suche den naechseten gueltigen Befehl
   while {$schleife == 1} {
      if {$DEBUG == "on"} {echo "right.Proc: while-Schleife 2"}
      # Ende des Befehls suchen
      set address [examine -time $time_now $minideb(TIME_UNIT) -hex $minideb(pc-path)]
      set address_tmp [examine -time [expr $time_now + $minideb(CLK_PERIOD)] $minideb(TIME_UNIT) -hex
$minideb(pc-path)]
      if {$address_tmp == "No_Data"} {
```

```
            .mainFrame.label_message configure -bg DimGray -fg red -text "End of Simulation reached."
            set minideb(auto_step_right) 0
            .mainFrame.frame_toolbar.autostep_right configure -background grey -activebackground khaki2
            .mainFrame.frame_toolbar.label_info configure -text ""
            return
         }
         set i 1
         while {$address == $address_tmp} {
            if {$DEBUG == "on"} {echo "right.Proc: while-Schleife 3"}
            incr i
            set address_tmp [examine -time [expr $time_now + $i * $minideb(CLK_PERIOD)] $mini-
deb(TIME_UNIT) -hex $minideb(pc-path)]
         }
         set time_now_tmp [expr $time_now + $i * $minideb(CLK_PERIOD)]
         # Befehl ist 16 Bit breit?
         set instruction [examine -time $time_now $minideb(TIME_UNIT) -hex $minideb(instruction)]
         set digit [split $instruction ""]
         set k1 "[lindex $digit 0][lindex $digit 1]"
         if {$DEBUG == "on"} {echo "right.Proc: splitted instruction --> <$k1>"}
         if {$k1 == "1E"} {
            # Befehl ist 16 Bit breit, 1 Befehl weitergehen
            if {$DEBUG == "on"} {echo "right.Proc: 16 Bit Befehl"}
            set time_now $time_now_tmp
         } else {
            set read_time [expr $time_now_tmp - 0.5 * $minideb(CLK_PERIOD)]
            if {$DEBUG == "on"} {echo "right.Proc:time_now --> $time_now; time_now_tmp -->
$time_now_tmp; read_time --> $read_time"}
            if {$DEBUG == "on"} {echo "en_pipe --> [examine -time $read_time $minideb(TIME_UNIT) -bin
$minideb(en_pipe)]; \
                                    res_pipe --> [expr ![examine -time $read_time $minideb(TIME_UNIT)
-bin $minideb(res_pipe)]]"}
            if {[examine -time $read_time $minideb(TIME_UNIT) -bin $minideb(en_pipe)] && ![examine -time
$read_time $minideb(TIME_UNIT) -bin $minideb(res_pipe)]} {
               # naechsten gueltigen Befehl gefunden
               set schleife 0
            } else {
               if {$DEBUG == "on"} {echo "right.Proc: Der Befehl ist ungueltig!"}
```

```
            # Befehl wird nicht ausgefuehrt, da
            # - pipe enable nicht gesetzt ist
            # - pipe reset gesetzt ist
            # 1 Befehl weiter gehen
            set time_now $time_now_tmp
         }
      }
   }
   set address_int [examine -time $time_now $minideb(TIME_UNIT) -hex $minideb(pc-path)]
   if {[catch {.$minideb(wave_name).tree right -value 'h$address_int 1} result] !=0} {
      # Wenn PC nicht markiert ist, deaktiviere Auto-Step und zeige eine Fehlermeldung an
      set minideb(auto_step_right) 0
      .mainFrame.frame_toolbar.autostep_right configure -background grey -activebackground khaki2
      .mainFrame.frame_toolbar.label_info configure -text ""
      notice_show "$result \nPlease select pc in the wave-Window!"
   }
   trace.Proc
}
```

Dabei wird vereinfacht wie folgt vorgegangen (siehe FIG 2):

- Zum momentanen Simulationszeitpunkt wird der Wert des Programmcounters 36 (pc, Programmcounter = Programmzähler) ermittelt. Der momentan ausgeführte Befehl ist in FIG 2 mit dem Bezugszeichen 31 gekennzeichnet)
- Der Beginn des nächsten Befehles wird gesucht. Der Cursor wird solange um Vielfache der Taktperiode weitergeschoben, bis eine Änderung des Programmzählers auftritt.
- Es wird überprüft, ob der nun erreichte Befehl ausgeführt wird.
  - Handelt es sich um einen Befehl, welcher nicht ausgeführt wird, so wird der übernächste Befehl getestet (Wiederholung solange, bis ein ausgeführter Befehl erreicht ist oder das Simulationsende erreicht ist). Im Beispielsfall gemäß FIG 2 sind die nicht ausgeführten, d. h. ungültigen Befehle mit dem Bezugszeichen 32 gekennzeichnet.
  - Wird der erreichte Befehl ausgeführt, dann wird der Cursor im Waveform-Fenster (entspricht dem zweiten Teilbereich 12 des Anzeigemittels 14 gemäß FIG 3) zu diesem Zeitpunkt platziert. Im Beispielsfall gemäß FIG 2 ist der nächste ausgeführte, d. h. gültige Befehl mit dem Bezugszeichen 33 gekennzeichnet.

Die Ermittlung des gültigen Befehls erfolgt prozessorspezifisch. Es genügt bei modernen Prozessorarchitekturen üblicherweise nicht, den Programcounter 36 zu verfolgen, sondern es sind zusätzliche Signale 4, 5 auszuwerten, die angeben, ob der momentane Befehl gültig ist (z. B. das der Waveform 34 zugrundeliegende Signal). Wenn ein gültiger Befehl ermittelt wurde, so erfolgt die Markierung 20 des momentan ausgeführten Befehls im angezeigten Listing im ersten Teilbereich 11 des Anzeigemittels 14 und es werden die Registerwerte im dritten Teilbereich 13 des Anzeigemittels 14 angezeigt (siehe folgenden Ausschnitt eines Listings).

```
proc trace.Proc {} {
  global progadr_alt
  global minideb
  global DEBUG
  if {$minideb(configure_done) == 1} {
    if {$minideb(cursor_move_enable) == 1} {
      DisableCursorMove.Proc
    }
    # Programmzaehler auslesen
    set timenow  [lindex [getactivecursortime] 0]
    # aktuelle Programmadresse lesen
    if {$DEBUG == "on"} {echo "trace.Proc: timenow eingelesen"}
    set progadr      [string tolower [examine -time $timenow $minideb(TIME_UNIT) -hex $minideb(pc-
path)]]
    if {$DEBUG == "on"} {echo "trace.Proc: progadr gelesen <$progadr>"}
    # Programmadresse ist undefiniert (Anfang von der Simulation) ?
    if {![regexp {[0-9a-fA-F]+} $progadr]} {
      set progadr 0
    }
    # gelesenen Programmcounter mit 2 multiplizieren, um die Programmadresse zu erhalten (bei NIOS)
    # vorangestellte Nullen werden abgeschnitten
    # da mit hex-Zahlen nicht gerechnet werden kann, wird auf eine Dezimalzahl umgerechnet,
    # das Ergebnis mit 2 multipliziert (bei NIOS, 1 bei KRISC) und anschließend wieder in eine
    # hex-Zahl konvertiert.
    set progadr_int  [hex2int $progadr]
    set progadr_int  [expr $minideb(adressmultiplikator)*$progadr_int]
    set progadr      [int2hex $progadr_int]

    SearchCodeLine.Proc $progadr
    GetRegisterValues_$minideb(mode).Proc
    set progradr_alt $progadr
  } else {
    bell
    .mainFrame.label_message configure -bg DimGray -fg red -text "Before tracing you have to configu-
re the program."
  }
}
```

Um den simulierten Ablauf eines Programms 3 auf einem Prozessor 7 zu verfolgen, gab es bisher keine zufriedenstellenden Möglichkeiten. Die manuelle Verfolgung des Programmablaufes anhand der von einem Simulator ausgegebenen Waveform und einem Ausdruck eines Listfiles hat den Nachteil, dass bei den üblicherweise verwendeten Prozessoren (Pipelines, Caches, u. Ä.) eine Programmverfolgung sehr aufwendig und fehlerträchtig wird. Bei einer Verfolgung des Programmablaufes anhand eines Disassemblerausdruckes muss ein Benutzer keine direkte Auswertung der Waveform vornehmen. Es wird nur eine Liste der vom Prozessor bearbeiteten Befehle ausgegeben. Somit fehlt aber der direkte Bezug zum eigentlichen Programm und damit werden auch die im Quelltext eingefügten Kommentare nicht angezeigt. Beim Verwenden eines Debuggers, der auf ein abstrahiertes C-Modell aufsetzt, ergeben sich im Wesentlichen zwei Nachteile. Zum einen benötigt man ein C-Modell für den eingesetzten Prozessor, welches normalerweise nicht vorliegt und somit aufwendig erstellt werden muss. Zum anderen stellt das C-Modell des Prozessors eine erneute Beschreibung des Prozessors dar und es ist nicht ausgeschlossen, dass der tatsächliche Prozessor ein vom C-Modell abweichendes Verhalten aufweist. Da bei dem hier vorgeschlagenen Verfahren kein C-Modell für den Prozessor 7 benötigt wird, ist zum einen gewährleistet, dass der gleiche Prozessor 7 simuliert wird, der auch in der Schaltung 6 implementiert ist. Zum anderen hält sich der Aufwand zum Anpassen des Debuggers 19 an verschiedene Prozessoren 7 bzw. Prozessortypen in Grenzen.

Zusammengefasst betrifft die Erfindung somit ein System und ein Verfahren zur Verknüpfung und Darstellung von Signalen 4, 5 einer Vorrichtung zur Hardware-Simulation 1 und Elementen 15 eines Listings 2 eines Programms 3 sowie ein Fehlersuchwerkzeug. Um eine gemeinsame Darstellung der Signale der Vorrichtung zur Hardware-Simulation und der Elemente des Listings des Programms zu ermöglichen, wird vorgeschlagen, dass die Vorrichtung zur Hardware-Simulation 1 das Verhalten einer Schaltung 6 mit einem Prozessor 7, einem Programmspeicher 8,

welcher den Programmcode 9 des Programms 3 enthält, und applikationsspezifischen Hardwarekomponenten 10 simuliert und Signale 4, 5 als Ergebnis der Simulation erzeugt, dass die Elemente 15 des Listings 2 des Programms 3 mit den bei der simulierten Ausführung des im Programmspeicher 8 enthaltenen, mit diesen Elementen 15 korrespondierenden Programmcodes 9 ' erzeugten Signalen 4, 5 verknüpft werden und dass die Elemente 15 des Listings 2 des Programms 3 in einem ersten Teilbereich 11 eines grafischen Anzeigemittels 14 und die Signale 4, 5 in einem zweiten Teilbereich 12 des Anzeigemittels 14 darstellbar sind.

Im Folgenden werden der technische Hintergrund der Erfindung und verwendete Begriffe näher erläutert.

Die beschriebenen Schaltungen werden z. B. in sogenannten eingebetteten Systemen (gebräuchlicher ist der entsprechende englische Begriff "embedded systems") eingesetzt. Beispiele für Programmiersprachen für den Einsatz in eingebetteten Systemen sind C, Assembler, C++ und Java. Diesen Sprachen gemeinsam ist der Einsatz eines Compilers, der eine schrittweise Vorgehensweise bei der Codeentwicklung vorschreibt, die sogenannten edit-compile-load-debug Zyklen. Debugger genannte Fehlersuchwerkzeuge werden allgemein verwendet, um Fehler in der Software-Programmierung aufzuspüren. Da die meisten Software-Entwicklungssysteme hierfür eine integrierte Entwicklungsumgebung zur Verfügung stellen, können Debugger relativ einfach den Programmcode verändern und mit Einzelschritten oder gesetzten Kontrollpunkten auf ihrem Zielsystem überprüfen. Man kann damit ein Programm kontrolliert ausführen, also das Programm Zeile für Zeile ausführen und die Werte von Variablen abfragen und ändern. Software Debugger bieten folgende Basisfunktionen an:

- Einzelschritt-Ausführung von Assembler- und Hochsprachencode

- Setzen von Haltepunkten (Breakpoints) an definierten Stellen in Hochsprache oder Maschinencode, an denen das Programm vorübergehend angehalten wird
- Anzeigen von Variablenwerten, logischen Ausdrücken, Speicheradressen und CPU-Registern

TCL ist eine plattformübergreifende Script-Programmiersprache. Durch die Kombination aus Textverarbeitungs-, Dateibearbeitungs- und Systemsteuerungsfunktionen ist TCL für diesen Zweck optimiert. EDA-Tools (EDA = Electronic Design Automation) verwenden TCL häufig zusammen mit dem Grafik-Toolkit TK, um eine flexible und plattformunabhängige grafische Benutzeroberfläche zu bieten. Hierzu gehört beispielsweise ModelSim.

Im Bereich der eingebetteten Systeme hat der parallele Entwurf von Hardware und Software den klassischen, rein sequentiellen Entwurfsablauf weitgehend abgelöst. Eingebettete Systeme spielen eine dominierende Rolle in der Automatisierungstechnik. Basis ihres Erfolgs ist die exponentiell steigende Komplexität integrierter Schaltungen und die damit wachsenden Möglichkeiten für neue Systemfunktionen, die zunehmend in Software realisiert werden. Die daraus resultierende Systemkomplexität verlangt bei gleichzeitig absinkenden Entwicklungszeiten eine drastische Erhöhung der Entwurfsproduktivität. Diese Erhöhung ist nur durch Entwurfsautomation und systematische Wiederverwendung von Systemfunktionen erreichbar. Ein wesentlicher Schritt ist die Integration von Hardware- und Softwareentwurf, der Hardware/Software-Coentwurf.

Hardware- und Softwareentwurf beginnen oft vor der Vollendung der Systemarchitektur oder gar vor der endgültigen Festlegung der Spezifikation. Systemarchitekten, Anwender und Kunden oder Marketingexperten entwickeln gemeinsam Anforderungsdefinition und Spezifikation. Der Systemarchitekt entwickelt daraus ein System kooperierender Systemfunktionen als Grundlage eines anschließenden, parallelen Entwurfs von Hardware und

Software. An dieser Stelle wird auch die Grobarchitektur der Zielhardware festgelegt. Der Hardware-/Software-Schnittstellenentwurf erfordert eine Beteiligung von Hardware- und Softwareentwicklern. Die Integration der Hardware und Softwarekomponenten und der Test des integrierten Systems folgt als letzter Schritt. In allen Phasen führen Abweichungen von erwarteten Entwurfsergebnissen oder Änderungen der Spezifikation zu einer Wiederholung von Entwurfsschritten. Ein zentrales Problem im Entwurfsprozess ist die Überwachung und Integration des parallelen Hardware- und Software-Entwurfs. Eine frühe Fehlererkennung erfordert die Kontrolle von Konsistenz und Korrektheit, die um so aufwendiger wird, je detaillierter der Entwurf ausgearbeitet ist.

Bei der Hardware-Software Cosimulation wird die Ausführung der Software auf der (Prozessor-)Hardware zusammen mit applikationsspezifischen Hardwarekomponenten simuliert. Da eine Simulation mit hohem Detaillierungsgrad, wie er im Hardwareentwurf üblich ist, zu langsam für eine praktisch nutzbare Softwaresimulation ist, werden üblicherweise abstrakte Prozessormodelle benötigt. Zu diesem Zweck werden die Prozessoren abstrakter ("auf einer höheren Abstraktionsebene") modelliert als die anderen Hardwarekomponenten. Dabei wird das Zeitverhalten des Prozessors nicht mehr taktgenau, d. h. für jeden Taktzyklus aufgeschlüsselt, dargestellt, sondern nur noch die Programme mit ihren Ein- und Ausgaben. Das Problem der Cosimulation liegt nun in der Kopplung der unterschiedlich abstrakten Modelle, derart dass eine hinreichende Genauigkeit der Simulation erreicht wird. Im ungünstigsten Fall greifen Prozessor und andere Hardwarekomponenten auf den gleichen Speicher zu. Eine genaue Modellierung erfordert in diesem Fall angepasste Speicher- und Busmodelle und spezielle Simulationstechniken. Ein Beispiel hierfür bietet der Cosimulator Seamless CVS der Firma Mentor Graphics, Wilsonville, Oregon, USA. Das Produkt Seamless CVS benutzt ein abstraktes Prozessormodell, das die Befehlsausführung nachbildet (ein sogenannter Instruction Set Simulator). Für den Speicher-

zugriff werden Busmodelle verwendet, deren Abstraktion abhängig vom gleichzeitigen Zugriff von Prozessor und Hardware ist. Speicher, auf die lediglich der Prozessor zugreift, werden folglich abstrakter modelliert als solche, in denen Konflikte auftreten können. Voraussetzung ist also die Verfügbarkeit einer Bibliothek von Modellen, die vom CAD-Anbieter oder dem Prozessorhersteller bezogen wird.

Ein abstrakterer Ansatz reduziert das Prozessormodell auf die reine Programmausführung auf dem PC oder der Workstation und modelliert lediglich das Interface mit Zeitverhalten. Die Kopplung der Software-Ausführung zum Hardwaremodell erfolgt dann über ein simulatorspezifisches Kommunikationsprotokoll, das der Entwickler in die Software einfügen muss. Für diese Modellierung sind nur Interface-Modelle erforderlich, was die Bibliotheksproblematik wesentlich vereinfacht. Andererseits wird das Zeitverhalten nur auf der Hardwareseite korrekt modelliert. Beispiel für einen solchen Cosimulator ist das Produkt Eagle der Firma Synopsys, Mountain View, Kalifornien, USA.

Patentansprüche

1. System zur Verknüpfung und Darstellung von Signalen (4, 5) einer Vorrichtung zur Hardware-Simulation (1) und Elementen (15) eines Listings (2) eines Programms (3),

- wobei die Vorrichtung zur Hardware-Simulation (1) das Verhalten einer Schaltung (6) mit einem Prozessor (7), einem Programmspeicher (8), welcher den Programmcode (9) des Programms (3) enthält, und applikationsspezifischen Hardwarekomponenten (10) simuliert und Signale (4, 5) als Ergebnis der Simulation erzeugt,
- wobei die Elemente (15) des Listings (2) des Programms (3) mit den bei der simulierten Ausführung des im Programmspeicher (8) enthaltenen, mit diesen Elementen (15) korrespondierenden Programmcodes (9) erzeugten Signalen (4, 5) verknüpft werden,
- wobei die Elemente (15) des Listings (2) des Programms (3) in einem ersten Teilbereich (11) eines grafischen Anzeigemittels (14) und die Signale (4, 5) in einem zweiten Teilbereich (12) des Anzeigemittels (14) darstellbar sind.

2. System nach Anspruch 1,
dadurch gekennzeichnet,
dass eine Markierung (20) eines Elements (15) des Listings (2) des Programms (3) im ersten Teilbereich (11) des grafischen Anzeigemittels (14) und eine Markierung (21) der mit diesem Element (15) verknüpften Signale (4, 5) im zweiten Teilbereich (12) des Anzeigemittels (14) vorgesehen ist.

3. System nach Anspruch 1 oder 2,
dadurch gekennzeichnet,
dass ein dritter Teilbereich (13) des grafischen Anzeigemittels (14) zur Darstellung mindestens eines Teils der Signale (4, 5) vorgesehen ist.

4. System nach einem der vorhergehenden Ansprüche,
dadurch gekennzeichnet,

dass die Schaltung (6) mit dem Prozessor (7), dem Programmspeicher (8) und den applikationsspezifischen Hardwarekomponenten (9) in einer Hardware-Beschreibungssprache beschrieben sind.

5. System nach einem der vorhergehenden Ansprüche, dadurch gekennzeichnet, dass Mittel zum Anpassen des Systems an verschiedene Prozessortypen vorgesehen sind.

6. Verfahren zur Verknüpfung und Darstellung von Signalen (4, 5) einer Vorrichtung zur Hardware-Simulation (1) und Elementen (15) eines Listings (2) eines Programms (3),

- wobei die Vorrichtung zur Hardware-Simulation (1) das Verhalten einer Schaltung (6) mit einem Prozessor (7), einem Programmspeicher (8), welcher den Programmcode (9) des Programms (3) enthält, und applikationsspezifischen Hardwarekomponenten (10) simuliert und Signale (4, 5) als Ergebnis der Simulation erzeugt,
- wobei die Elemente (15) des Listings (2) des Programms (3) mit den bei der simulierten Ausführung des im Programmspeicher (8) enthaltenen, mit diesen Elementen (15) korrespondierenden Programmcodes (9) erzeugten Signalen (4, 5) verknüpft werden,
- wobei die Elemente (15) des Listings (2) des Programms (3) in einem ersten Teilbereich (11) eines grafischen Anzeigemittels (14) und die Signale (4, 5) in einem zweiten Teilbereich (12) des Anzeigemittels (14) dargestellt werden.

7. Verfahren nach Anspruch 6, dadurch gekennzeichnet, dass ein Element (15) des Listings (2) des Programms (3) im ersten Teilbereich (11) des grafischen Anzeigemittels (14) und die mit diesem Element (15) verknüpften Signale (4, 5) im zweiten Teilbereich (12) des Anzeigemittels (14) markiert werden.

8. Verfahren nach Anspruch 6 oder 7,
dadurch gekennzeichnet,
dass in einem dritten Teilbereich (13) des grafischen Anzeigemittels (14) mindestens ein Teil der Signale (4, 5) dargestellt wird.

9. Verfahren nach einem der Ansprüche 6 bis 8,
dadurch gekennzeichnet,
dass die Schaltung (6) mit dem Prozessor (7), dem Programmspeicher (8) und den applikationsspezifischen Hardwarekomponenten (9) in einer Hardware-Beschreibungssprache beschrieben sind.

10. Verfahren nach einem der Ansprüche 6 bis 9,
dadurch gekennzeichnet,
dass das Verfahren an verschiedene Prozessortypen anpassbar ist.

11. Fehlersuchwerkzeug zur Verknüpfung und Darstellung von Signalen (4, 5) einer Vorrichtung zur Hardware-Simulation (1) und Elementen (15) eines Listings (2) eines Programms (3),

- wobei die Vorrichtung zur Hardware-Simulation (1) das Verhalten einer Schaltung (6) mit einem Prozessor (7), einem Programmspeicher (8), welcher den Programmcode (9) des Programms (3) enthält, und applikationsspezifischen Hardwarekomponenten (10) simuliert und Signale (4, 5) als Ergebnis der Simulation erzeugt,
- wobei das Fehlersuchwerkzeug Mittel zur Verknüpfung der Elemente (15) des Listings (2) des Programms (3) mit den bei der simulierten Ausführung des im Programmspeicher (8) enthaltenen, mit diesen Elementen (15) korrespondierenden Programmcodes (9) erzeugten Signalen (4, 5) aufweist,

wobei die Elemente (15) des Listings (2) des Programms (3) in einem ersten Teilbereich (11) eines grafischen Anzeigemittels (14) und die Signale (4, 5) in einem zweiten Teilbereich (12) des Anzeigemittels (14) darstellbar sind.

FIG 1

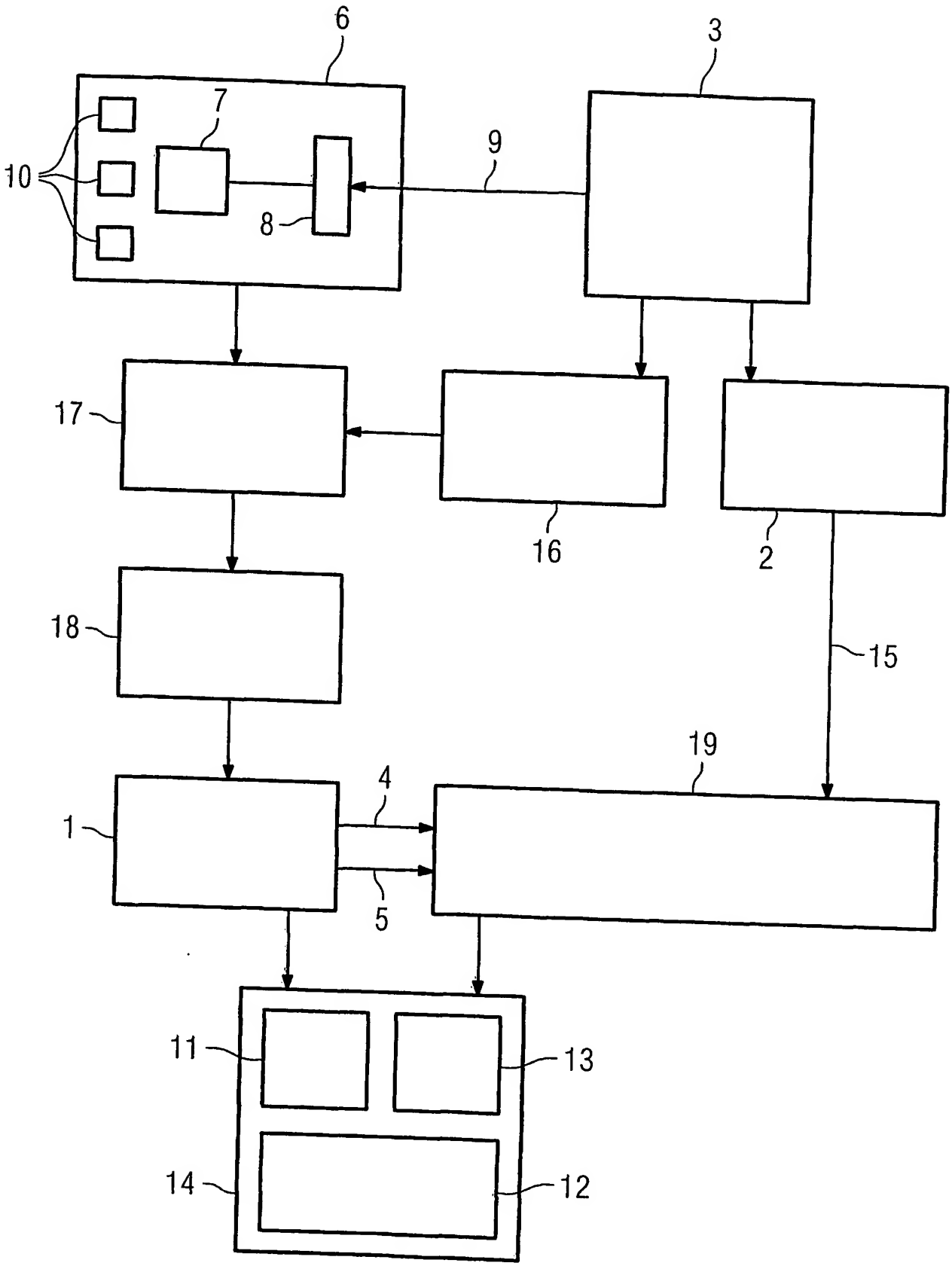


FIG 2

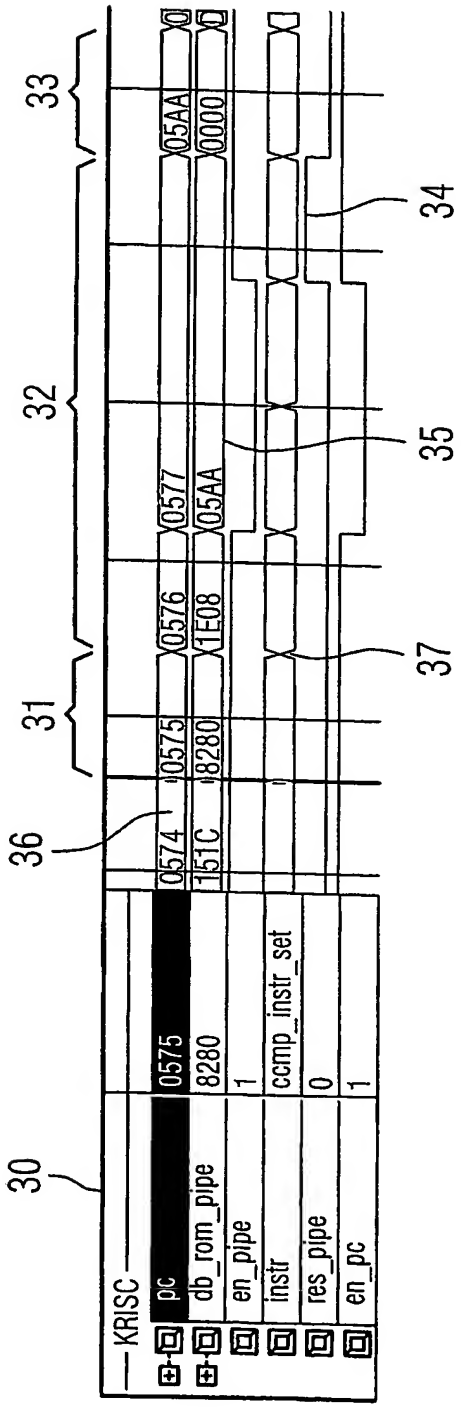
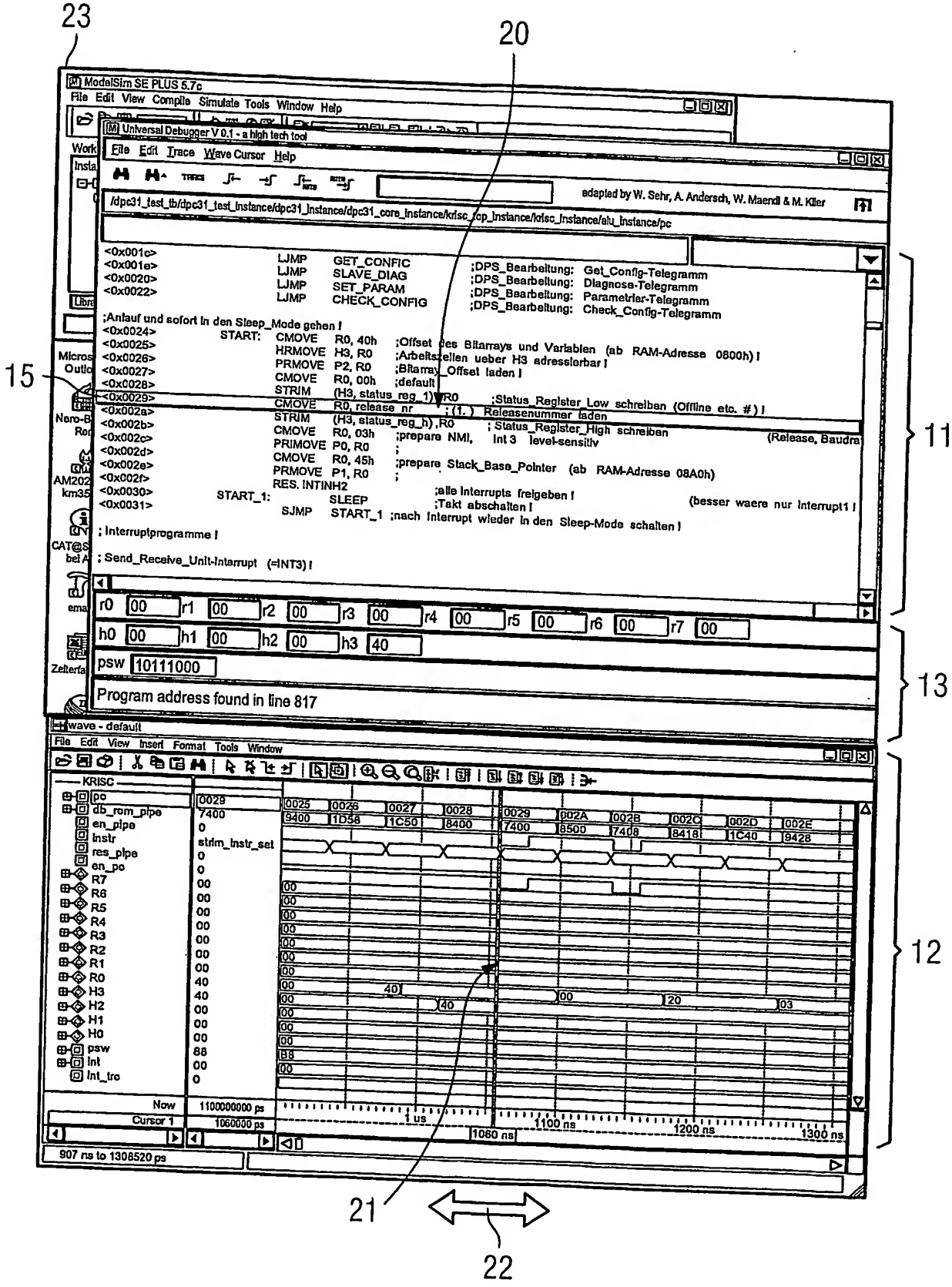

KRISC
pc
db_rom_pipe
en_pipe
instr
res_pipe
en_pc
0575
8280
1
ccmp_instr_set
0
1
0574
151C
0575
8280
0576
1E08
0577
05AA
05AA
0000
30
31
32
33
34
35
36
37

FIG 3


23
20
15
11
13
12
21
22
ModelSim SE PLUS 5.7c
File Edit View Compile Simulate Tools Window Help
Universal Debugger V 0.1 - a high tech tool
File Edit Trace Wave Cursor Help
adapted by W. Sehr, A. Andersch, W. Maendl & M. Kiler
/dpc31_test_tb/dpc31_test_instance/dpc31_instance/dpc31_core_instance/krisc_top_instance/krisc_instance/alu_instance/pc
;Anlauf und sofort in den Sleep_Mode gehen !
;Interruptprogramme !
; Send_Receive_Unit-Interrupt (=INT3) !
r0 00 r1 00 r2 00 r3 00 r4 00 r5 00 r6 00 r7 00
h0 00 h1 00 h2 00 h3 40
psw 10111000
Program address found in line 817
Wave - default
File Edit View Insert Format Tools Window
KRISC
pc
db_rom_pipe
en_pipe
instr
res_pipe
en_pc
R7
R6
R5
R4
R3
R2
R1
R0
H3
H2
H1
H0
psw
int
int_trc
Now
Cursor 1
907 ns to 1308520 ps

INTERNATIONAL SEARCH REPORT

International Application No
PCT/EP2004/004991

A. CLASSIFICATION OF SUBJECT MATTER
IPC 7 G06F17/50

According to International Patent Classification (IPC) or to both national classification and IPC

B. FIELDS SEARCHED

Minimum documentation searched (classification system followed by classification symbols)
IPC 7 G06F

Documentation searched other than minimum documentation to the extent that such documents are included in the fields searched

Electronic data base consulted during the international search (name of data base and, where practical, search terms used)
EPO-Internal, INSPEC, COMPENDEX

C. DOCUMENTS CONSIDERED TO BE RELEVANT

Category °	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
X	WO 01/95161 A (VIRTIO CORP) 13 December 2001 (2001-12-13) paragraph '0085! paragraph '0094! paragraph '0097! paragraph '0102! - paragraph '0103! paragraph '0116! paragraph '0142! paragraph '0147! - paragraph '0148! paragraph '0157! - paragraph '0158! paragraph '0171! paragraph '0182! ----- -/--	1-11

[X] Further documents are listed in the continuation of box C.

[X] Patent family members are listed in annex.

° Special categories of cited documents :

"A" document defining the general state of the art which is not considered to be of particular relevance
"E" earlier document but published on or after the international filing date
"L" document which may throw doubts on priority claim(s) or which is cited to establish the publication date of another citation or other special reason (as specified)
"O" document referring to an oral disclosure, use, exhibition or other means
"P" document published prior to the international filing date but later than the priority date claimed

"T" later document published after the international filing date or priority date and not in conflict with the application but cited to understand the principle or theory underlying the invention
"X" document of particular relevance; the claimed invention cannot be considered novel or cannot be considered to involve an inventive step when the document is taken alone
"Y" document of particular relevance; the claimed invention cannot be considered to involve an inventive step when the document is combined with one or more other such documents, such combination being obvious to a person skilled in the art.
"&" document member of the same patent family

Date of the actual completion of the international search	Date of mailing of the international search report
3 September 2004	21/09/2004

Name and mailing address of the ISA	Authorized officer
European Patent Office, P.B. 5818 Patentlaan 2 NL – 2280 HV Rijswijk Tel. (+31-70) 340-2040, Tx. 31 651 epo nl, Fax: (+31-70) 340-3016	Radev, B

INTERNATIONAL SEARCH REPORT

International Application No
PCT/EP2004/004991

C.(Continuation) DOCUMENTS CONSIDERED TO BE RELEVANT

Category °	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
X	GUERRA L ET AL: "Cycle and phase accurate DSP modeling and integration for HW/SW co-verification" DESIGN AUTOMATION CONFERENCE, 1999. PROCEEDINGS. 36TH NEW ORLEANS, LA, USA 21-25 JUNE 1999, PISCATAWAY, NJ, USA,IEEE, US, 21 June 1999 (1999-06-21), pages 964-969, XP010344032 ISBN: 1-58113-092-9 abstract page 964, left-hand column, line 1 - right-hand column, line 11 page 967, right-hand column, line 11 - page 969, left-hand column, line 20 figure 7 -----	1-11
X	LE MARREC P ET AL: "Hardware, software and mechanical cosimulation for automotive applications" RAPID SYSTEM PROTOTYPING, 1998. PROCEEDINGS. 1998 NINTH INTERNATIONAL WORKSHOP ON LEUVEN, BELGIUM 3-5 JUNE 1998, LOS ALAMITOS, CA, USA,IEEE COMPUT. SOC, US, 3 June 1998 (1998-06-03), pages 202-206, XP010283205 ISBN: 0-8186-8479-8 abstract page 204, left-hand column, line 1 - page 205, left-hand column, line 14 figure 5 -----	1-11

INTERNATIONAL SEARCH REPORT

Information on patent family members

International Application No
PCT/EP2004/004991

Patent document cited in search report	Publication date	Patent family member(s)	Publication date
WO 0195161 A	13-12-2001	AU 6666001 A	17-12-2001
		WO 0195161 A2	13-12-2001
		US 2002059054 A1	16-05-2002

INTERNATIONALER RECHERCHENBERICHT

Internationales Aktenzeichen
PCT/EP2004/004991

A. KLASSIFIZIERUNG DES ANMELDUNGSGEGENSTANDES
IPK 7 G06F17/50

Nach der Internationalen Patentklassifikation (IPK) oder nach der nationalen Klassifikation und der IPK

B. RECHERCHIERTE GEBIETE

Recherchierter Mindestprüfstoff (Klassifikationssystem und Klassifikationssymbole)
IPK 7 G06F

Recherchierte aber nicht zum Mindestprüfstoff gehörende Veröffentlichungen, soweit diese unter die recherchierten Gebiete fallen

Während der internationalen Recherche konsultierte elektronische Datenbank (Name der Datenbank und evtl. verwendete Suchbegriffe)
EPO-Internal, INSPEC, COMPENDEX

C. ALS WESENTLICH ANGESEHENE UNTERLAGEN

Kategorie°	Bezeichnung der Veröffentlichung, soweit erforderlich unter Angabe der in Betracht kommenden Teile	Betr. Anspruch Nr.
X	WO 01/95161 A (VIRTIO CORP) 13. Dezember 2001 (2001-12-13) Absatz '0085! Absatz '0094! Absatz '0097! Absatz '0102! - Absatz '0103! Absatz '0116! Absatz '0142! Absatz '0147! - Absatz '0148! Absatz '0157! - Absatz '0158! Absatz '0171! Absatz '0182! ----- -/--	1-11

[X] Weitere Veröffentlichungen sind der Fortsetzung von Feld C zu entnehmen

[X] Siehe Anhang Patentfamilie

° Besondere Kategorien von angegebenen Veröffentlichungen :
"A" Veröffentlichung, die den allgemeinen Stand der Technik definiert, aber nicht als besonders bedeutsam anzusehen ist
"E" älteres Dokument, das jedoch erst am oder nach dem internationalen Anmeldedatum veröffentlicht worden ist
"L" Veröffentlichung, die geeignet ist, einen Prioritätsanspruch zweifelhaft erscheinen zu lassen, oder durch die das Veröffentlichungsdatum einer anderen im Recherchenbericht genannten Veröffentlichung belegt werden soll oder die aus einem anderen besonderen Grund angegeben ist (wie ausgeführt)
"O" Veröffentlichung, die sich auf eine mündliche Offenbarung, eine Benutzung, eine Ausstellung oder andere Maßnahmen bezieht
"P" Veröffentlichung, die vor dem internationalen Anmeldedatum, aber nach dem beanspruchten Prioritätsdatum veröffentlicht worden ist
"T" Spätere Veröffentlichung, die nach dem internationalen Anmeldedatum oder dem Prioritätsdatum veröffentlicht worden ist und mit der Anmeldung nicht kollidiert, sondern nur zum Verständnis des der Erfindung zugrundeliegenden Prinzips oder der ihr zugrundeliegenden Theorie angegeben ist
"X" Veröffentlichung von besonderer Bedeutung; die beanspruchte Erfindung kann allein aufgrund dieser Veröffentlichung nicht als neu oder auf erfinderischer Tätigkeit beruhend betrachtet werden
"Y" *Veröffentlichung von besonderer Bedeutung; die beanspruchte Erfindung* kann nicht als auf erfinderischer Tätigkeit beruhend betrachtet werden, wenn die Veröffentlichung mit einer oder mehreren anderen Veröffentlichungen dieser Kategorie in Verbindung gebracht wird und diese Verbindung für einen Fachmann nahellegend ist
"&" Veröffentlichung, die Mitglied derselben Patentfamilie ist

Datum des Abschlusses der internationalen Recherche	Absendedatum des internationalen Recherchenberichts
3. September 2004	21/09/2004
Name und Postanschrift der Internationalen Recherchenbehörde Europäisches Patentamt, P.B. 5818 Patentlaan 2 NL – 2280 HV Rijswijk Tel. (+31-70) 340-2040, Tx. 31 651 epo nl, Fax: (+31-70) 340-3016	**Bevollmächtigter Bediensteter** Radev, B

INTERNATIONALER RECHERCHENBERICHT

Internationales Aktenzeichen
PCT/EP2004/004991

C.(Fortsetzung) ALS WESENTLICH ANGESEHENE UNTERLAGEN

Kategorie°	Bezeichnung der Veröffentlichung, soweit erforderlich unter Angabe der in Betracht kommenden Teile	Betr. Anspruch Nr.
X	GUERRA L ET AL: "Cycle and phase accurate DSP modeling and integration for HW/SW co-verification" DESIGN AUTOMATION CONFERENCE, 1999. PROCEEDINGS. 36TH NEW ORLEANS, LA, USA 21-25 JUNE 1999, PISCATAWAY, NJ, USA,IEEE, US, 21. Juni 1999 (1999-06-21), Seiten 964-969, XP010344032 ISBN: 1-58113-092-9 Zusammenfassung Seite 964, linke Spalte, Zeile 1 - rechte Spalte, Zeile 11 Seite 967, rechte Spalte, Zeile 11 - Seite 969, linke Spalte, Zeile 20 Abbildung 7	1-11
X	LE MARREC P ET AL: "Hardware, software and mechanical cosimulation for automotive applications" RAPID SYSTEM PROTOTYPING, 1998. PROCEEDINGS. 1998 NINTH INTERNATIONAL WORKSHOP ON LEUVEN, BELGIUM 3-5 JUNE 1998, LOS ALAMITOS, CA, USA,IEEE COMPUT. SOC, US, 3. Juni 1998 (1998-06-03), Seiten 202-206, XP010283205 ISBN: 0-8186-8479-8 Zusammenfassung Seite 204, linke Spalte, Zeile 1 - Seite 205, linke Spalte, Zeile 14 Abbildung 5	1-11

INTERNATIONALER RECHERCHENBERICHT

Angaben zu Veröffentlichungen, die zur selben Patentfamilie gehören

Internationales Aktenzeichen
PCT/EP2004/004991

Im Recherchenbericht angeführtes Patentdokument	Datum der Veröffentlichung	Mitglied(er) der Patentfamilie	Datum der Veröffentlichung
WO 0195161 A	13-12-2001	AU 6666001 A	17-12-2001
		WO 0195161 A2	13-12-2001
		US 2002059054 A1	16-05-2002